VOL 1 (2017) NO 3

e-ISSN : 2549-9904
ISSN : 2549-9610

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

Model-Based Resource Utilization and Performance Risk Prediction using Machine Learning Techniques

Haitham A.M Salih[#], Hany H Ammar[*]

[#] *College of Graduate Studies, Sudan University of Science and Technology, Khartoum, Sudan*
[*] *Lane Department of Computer Science and Electrical Engineering, West Virginia University, West Virginia, USA*
E-mail: a_hytham@hotmail.com, hammar@wvu.edu

***Abstract*-The growing complexity of modern software systems makes the performance prediction a challenging activity. Many drawbacks incurred by using the traditional performance prediction techniques such as time consuming and inability to surround all software system when large scaled. To contribute to solving these problems, we adopt a model-based approach for resource utilization and performance risk prediction. Firstly, we model the software system into annotated UML diagrams. Secondly, performance model is derived from UML diagrams in order to be evaluated. Thirdly, we generate performance and resource utilization training dataset by changing workload. Finally, when new instances are applied we can predict resource utilization and performance risk by using machine learning techniques. The approach will be used to enhance work of human experts and improve efficiency of software system performance prediction. In this paper, we illustrate the approach on a case study. A performance training dataset has been generated, and three machine learning techniques are applied to predict resource utilization and performance risk level. Our approach shows prediction accuracy within 68.9 % to 93.1 %.**

Keywords **- Machine learning, Performance, Risk prediction, WEKA.**

I. INTRODUCTION

Software Performance considered to be the most important non-functional requirements especially for real time systems, aircraft system, medical system, and ecommerce system [1]. The Non-functional determine what software will do, as well as how the software will do it. In addition, non-functional requirements validation for the software so far does not get enough consideration in the practical real life of the software engineers. Running after the market is still ruling in the software development life cycle, however the quality of the final product suffers after being put the software in the business working environment [1].

Software Performance risk means undesirable action that restrain the software to giving extreme functionality under all probable environment circumstances [2]. Early detection of software performance metrics such as system response time, hardware utilization, and system throughput is a key step to manage risks of software system before going to implementation phase [3]. The software performance requirements can be categorized into two types:

- Time-related performance requirement, that means the completion time of a particular operation, must be less than a certain value.
- Resource-related performance requirement, that means the utilization of a specific device, must fall into a specific range.

Using current methods of software performance prediction such as simulation, guessing, and software engineer previous experience have inconvenient results especially on complex and large scale software. Instead, machine learning can provides methods, techniques, and tools that can help in solving many prediction problems for variety of software quality requirements such as maintainability, performance, and hardware configuration management [4].

The paper proposes a new approach for model-based resource utilization and performance risk prediction by using machine learning techniques. We used multiple-regression algorithm for resource utilization prediction, also we applied three machine learning algorithms such as K-Nearest Neighbor, Support Vector Machine, and Naïve Bayes to predict risk level. Firstly, we will model the system by using annotated Unified Modeling Language (UML) diagrams such as use case, sequence diagram, and deployment diagram. Secondly, we will map the annotated UML models into performance model - Queuing Network Model (QNM). Thirdly, we will solve the QNM model to generate large training dataset by changing workloads and number of users. Finally, we will apply the new instances as a

test sets on the machine learning to predict resource utilization of the servers and further to classify risks on the test sets as high, medium, or low risk. Unlike the previous work, the approach focuses mainly on software performance prediction at the early modeling stages of software development life cycle. Unlike, prediction after moving into implementation, which incurred high cost and many software changes.

This paper is organized as follows: Section 2 presents the material and method. Section 3 dedicated to results as well as the explanation of results. In Section 4, we finished the paper by the conclusion.

II. Material And Method

Machine learning algorithms can be potentially powerful tool to enhance resource utilization and software performance risk prediction. Normally, performance modeling tools such as Queuing Network Model, Petri Nets, and Markov Chain can only produces quantitative performance indices for the software models assessment. Furthermore, by collecting large training dataset the machine learning algorithms can easily analyzes, predicts and gives both quantitative and qualitative performance indices. These powerful features of machine learning will free the software engineers from the complexity process of performance prediction and focus more on other software development tasks.

A. Queuing Network Model (QNM)

Queuing network model is a mathematical model that used to evaluate performance of computer systems [3]. The fundamental parts of QNM are *queues* and *service centers*. A queue is a buffer same to any queuing system. A service center has a related queue including jobs to be served by that service center.

To get performance attributes for any individual service center, two kinds of information must be available. The average arrival time of jobs ***R***, and the average time consumed by service center ***S*** to perform one job. Based on this information following quality of services could be calculated:

- Utilization of each system component:
$$U_i = R * S_i \quad (1)$$
- Average of queue length:
$$q_i = u_i^2/(1-u_i) \quad (2)$$
- Average Response Time:
$$S_i/(1-u_i) \quad (3)$$
- Average population of the component:
$$p_i = u_i/(1-u_i) \quad (4)$$

From the above equations it is possible to make calculations for latency, throughput, and highly utilized service centers.
The population of the system is the sum of population of all components:

$$P = P_x + P_y + P_z \quad (5)$$

Where **x**, **y**, and **z** are the components of the system. In our paper we will use Java Modeling Tool (JMT). JMT is a free open source suite uses many algorithms for the exact, asymptotic and simulative analysis of queuing network models. [5].

B. Machine learning

Machine learning is the ability of the computer programs to capture or develop new knowledge from existing or non-existing instances for the purpose of enhancing the performance criteria [6]. Software engineers and researchers have been started using machine learning techniques in the area of quality of service classification and prediction. Moreover, machine learning has proved its efficiency to asset and optimizes model based performance prediction. We used Waikato Environment for Knowledge Analysis (WEKA) as a free tool for machine learning. The machine learning can be categorized into supervised and unsupervised learning.

1) *Supervised learning:* Supervised learning consists of algorithms that reason from externally supplied instances to produce general hypothesis which then make predictions about unknown instances. Also, with supervised learning there is existence of the outcome variable to orient the learning process. There are many machine learning algorithms for supervised learning such as Support Vector Machine (SVM), K-Nearest Neighbor, and Random Forests[7].
2) *Unsupervised learning:* In contrast to supervised learning where there is existence of the outcome variable to direct the learning process, unsupervised learning builds models from data without pre-defined example. This means no direction is available and learning must perform heuristically by the algorithm examining different training dataset.

C. Machine learning algorithms

There are various machine learning algorithms depending on the application domain; only four techniques, that is Multivariate regression, Naïve base, Multi perceptron, and K-Nearest Neighbor, will be discussed.

1) *Multivariate linear regression:* Multivariate linear regression (MLR) is the most commonly used technique for modeling the relationship between two or more independent and dependent variables by finding a linear question to observed data [8]. The general form of a MLR can be given be:

$$\hat{y_i} = a_0 + a_1 x_{i1} + \ldots + a_k x_{ik} \quad (6)$$
$$y_i = a_0 + a_1 x_{i1} + \ldots + a_k x_{ik} + e_i \quad (7)$$

where $x_{i1}, \ldots, x_{ik}$ are the independent variables, $a_0, \ldots, a_k$ the parameters to be evaluated, $\hat{y_i}$ the dependent variable to be predicted, y_i the actual value of the dependent variable, and e_i is the error in the prediction of the i^{th} case.

2) *Naïve Bayes:* Naïve Bayes models are simple models, treats all variables X_i are independent given a special variable C. The joint distribution is then given by

$$P(C, X_1, \dots, X_n) = P(C) \prod_{i=1}^{n} P(X_i|C) \qquad (8)$$

The univariate conditional distributions $P(X_i|C)$ can accept any form such as multinomial for discrete variables, Gaussian for continuous ones. When the variable C is observed in the training dataset, naïve Bayes can be used for classification, by assigning test set example $(X_1,\dots,X_n)$ to the class C with maximum $P(C|X_1,\dots,X_n)$ [9].

3) *K-Nearest Neighbor (KNN):* K-Nearest is one of the methods referred to as instance based learning which categorized as supervised learning algorithm [6][10]. KNN works by simply storing the training data set, and when a new instance is applied, a set of similar related instances that are neighbors is gathered from the training dataset set, also KNN used to classify the new instance. Classification is useful to take more than one neighbor into account and then referred to as k-nearest neighbor.

4) *Support Vector Machines (SVMs):* Support Vector machine (SVMs) finds separating hyper-planes between training datasets that maximize the margin and minimize the classification errors[6][11]. Margins sometimes known as "geometric margin" and defined as distance between the hyper planes dividing two classes and the nearest data points to the hyper planes. The SVM algorithm is able to work with both linearly and separable problems in classification and regression tasks.

From the literature, Ganapathi used machine learning on software performance prediction [12]. She proposed a machine learning technique to predict/optimize multi components, parallel system utilization and performance. The proposed technique gathers the correlation between a workload's pre-execution characteristics configuration parameters, and post-execution performance observations. Finally, the correlation has been used for performance prediction and optimization. To prove the methodology, she presents three cases on storage and computer-based parallel systems. The outcomes suggest the use of machine learning based performance modeling to improve the quality of system management decisions.

The above work is very useful representation of using statistical machine learning to predict the performance of software systems; however, the approaches focused on software systems that already designed and implemented not that are at the early modeling stage.

Dubach and et al. used machine learning technique to explore the good compiler architecture design [13]. He designed two performance models and applied them to increase the efficiency of searching the design space for micro architecture. Models predict performance metrics such as processor cycles, energy consumption, and the trade- off of the two characteristics.

Malhotra and et al. have employed machine learning to measure software maintainability. Number of the word "change" is observed over a period of three years on a dataset. The researchers proved that when using Naïve Bayesian algorithm the classification performs better than other machine learning algorithms [4].

Finally, Ipek and et al, used multilayer neural network, the network trained on input data collected from execution on targeted platform [14]. This approach is useful for predicting many aspects of performance and it takes full system complexity. The study focuses on the high performance parallel application SMG2000. The model has predicted performance within error 5% to 7% across a large, multidimensional parameter space.

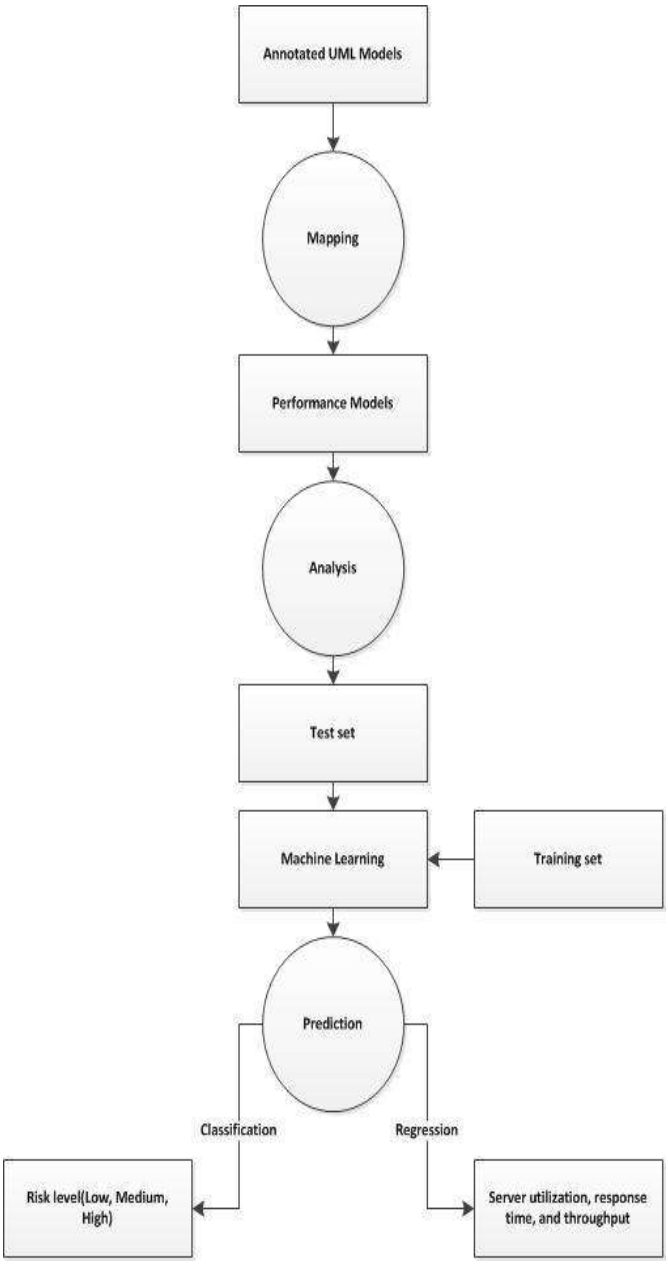

Fig. 1 Methodology of resource utilization and risk prediction

However, the work of Dubach, Malhotra, and Ipek applied on system that already designed and implemented, but they didn't start the prediction process of software performance from the early modeling stage. Our approach emphasizes on building and evaluating performance model, so that if the model gives reasonable performance indicators then we will continue and construct the model, otherwise the change will be on the model itself not on the software when it would be running.

UML profile for performance, Schedulability, and time has been announced by Object Management Group (OMG) as standard specification mechanism [15]. Starting from model-to-model transformation we have to take annotation tags and stereotypes proposed in the profile, and the ability to add more specifications. UML annotations concepts are enough to describe performance attributes of software systems. Moreover, extending the queuing theory with the machine learning is one of a new concept we have introduced in our paper.

The main contribution of the paper is to combine resource utilization and performance risk prediction from annotated UML models. The idea of our method is to combine the performance and the risk on UML software models. In real life, the exact quantitative performance prediction is not enough in specific situations, such as e-business systems where the delay in response time may lead to losing thousands of customers. In addition, software engineers looking for decisions on leveling the consequences of risk such as low, medium, or high risk.

We introduced a method to predict resource utilization and performance risk from the software system modeled with annotated UML diagrams. In this paper we considered performance risk as both requirements: *time related - resource related*, and we merged them. Moreover, we can predict resource utilization by formulating the problem as statistical and we apply linear regression algorithm. We also, predict the risk level as (low, medium, or high) by formulating the problem as classification prediction problem. To illustrate the method clearly we presented the steps in Fig.1. Next we illustrated the methodology by using a complete case study.

To prove the approach we will apply our method on Hospital system Fig. 2 as a case study. The system under analysis is part of the IT system of a hospital. The information system manages the patient history, e.g. the arrival date in the hospital, past diseases, or remarks related to his recovery as daily measured temperature.

The medicines issues are managed centrally: doctors insert the required quantity for a certain patient and the pharmacy of the hospital make orders for the needed amount of a needed medicine. Doctors insert data of patient's visits and matters. The nurses insert degrees of temperature, pressure. They also get from the system the required medicine for the patient. Moreover, there are functionalities that give the possibility to create a new record for a patient and get a file searching for his name or patient's number.

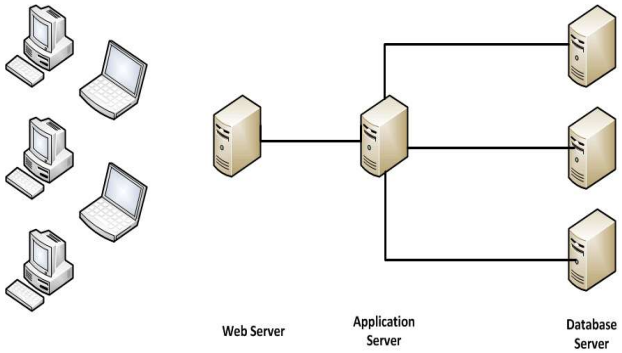

Fig. 2 Hospital System Architecture as 3-tier architecture

We follow the steps of the method and apply these steps on the case study.

A. Develop UML Models

Use case diagram: Each actor in a Use Case diagram may represent a stream of requests arriving at the system. There may be unlimited sequence of requests (open workload), or fixed population of users requiring service from the system service from the system (closed workload) [16].

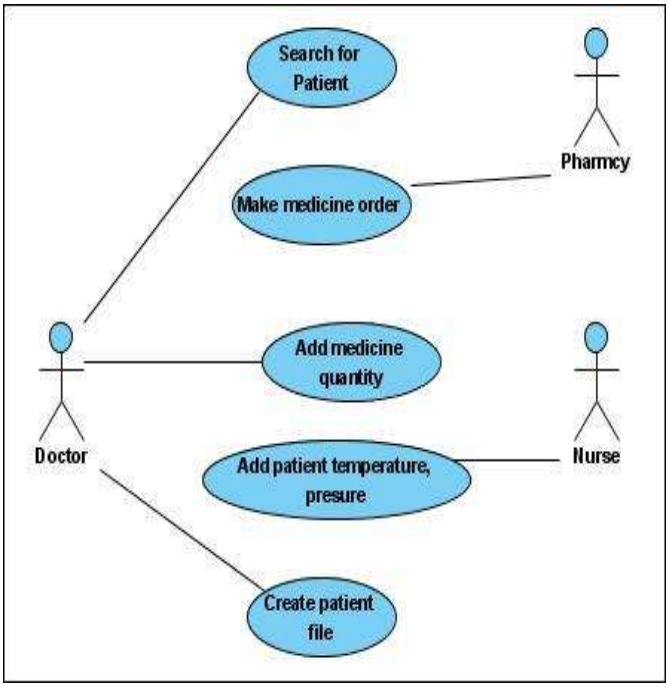

Fig. 3 Hospital System Use Case Diagram

Sequence diagram: Shows interactions consisting of a set of objects and the messages sent and received by those objects. Sequence diagram address the dynamic behaviour of a system with special emphasis on the chronological ordering of messeages.

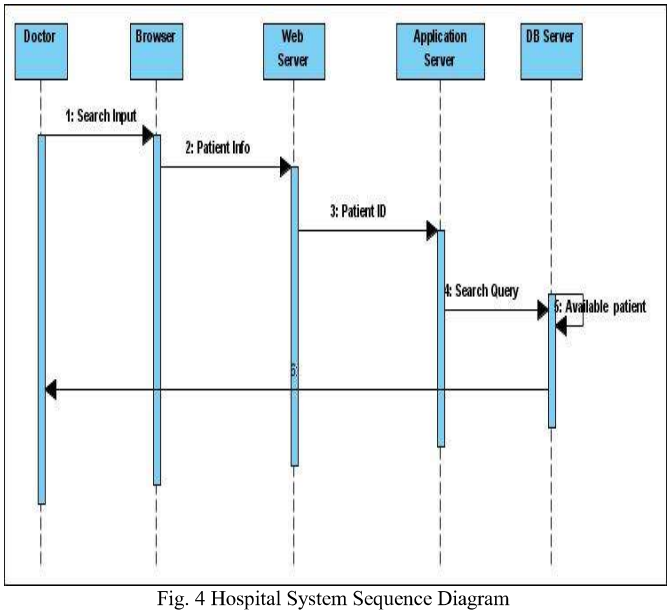

Fig. 4 Hospital System Sequence Diagram

Deployment diagram: Deployment diagrams are used to model the physical resources available in the system. Each resource is represented by a node in the deployment diagrams.

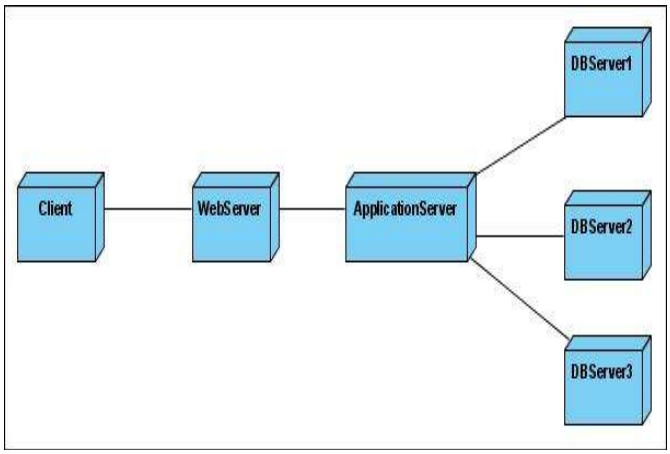

Fig. 5 Hospital System Deployment Diagram

B. Extract the Performance Model -QNM

The previous UML diagrams used to extract the QNM model. QNM provides the modeler with a wide range of quantitative evaluation of performance indices such as throughput, residence time, response time, utilization, and system power. We used JMT to run our hospital system model and generate a dataset of results. The training dataset extracted by changing the workload (number of users and type of load).

The dataset contains 194 instances and 6 attributes. The attributes are number of users, system throughput, system response time, web server utilization, application server utilization, and databases servers' utilization.

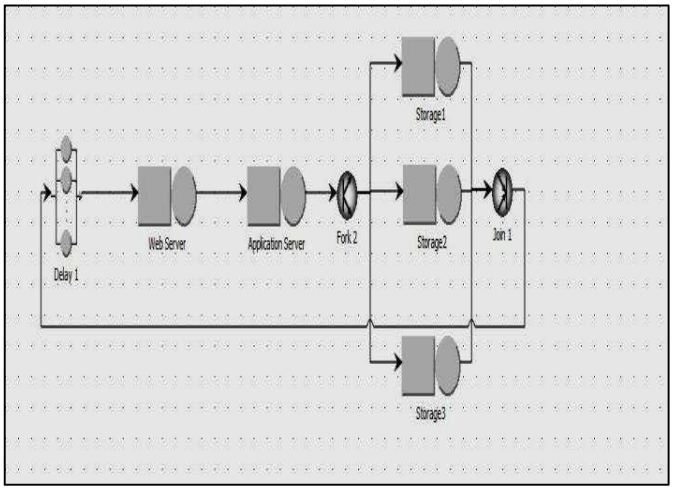

Fig. 6 Hospital System modeled as QNM Model

Fig. 6 represents the system under studying using JMT. The first element represents the number of users ***N*** and the model evaluated separately for each value of ***N***. The interaction of the users with the system is provided by a web application with a browser on client side and a typical 3 tiers on server. The requests are submitted to a web server, in case of a static request response immediately passed back to the client, otherwise the web server communicates with an application server that executes queries on a backend database and generated data are passed back to the web server. At last, clients receive the dynamically request. The storage consists of 3 database servers in parallel. The redirection of a job to one of the storage servers is made by a load balancer in a random fashion.

We subdivide the requests arriving at the system into two groups: the request for a database search "Heavy Load" and the request a database modification "Light Load". In Table 1 we reported the service demands for each station in the system and for each class.

TABLE 1
SERVICE DEMAND IN MILISECONDS FOR STATION IN THE SYSTEM

	Light Load	**Heavy Load**
Web Server	1.40	1.10
App Server	2.10	1.50
Storage 1	1.10	2.90
Storage 2	1.20	2.70
Storage 3	1.10	2.80

In Fig.7 we plotted global response time in function of the number of customers in the system keeping constant the mix of the two classes. Depending to the theory for high values of the number of customers, the response time grows linearly.

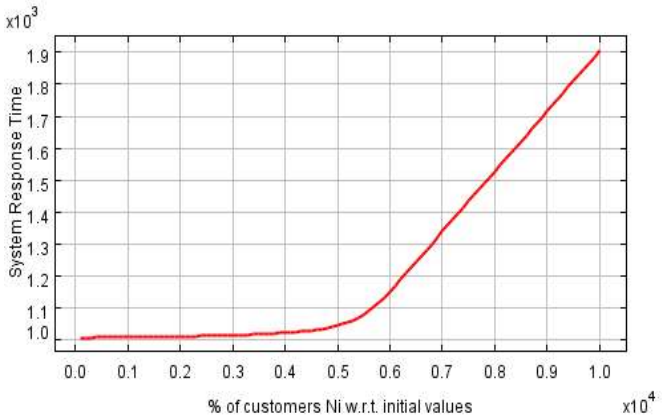

Fig. 7 System response time

In Fig.8 we reported utilization of each station with rising number of customers in the system. The upper line refers to the application server, the lower line refers to the web server and the other three lines represent the storage servers.

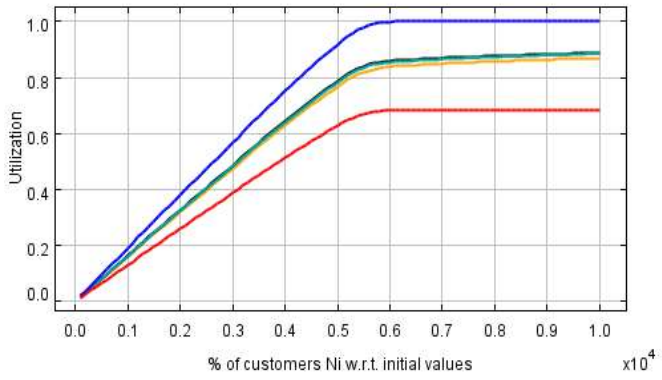

Fig. 8 System utilization

In Fig.9 we plotted global system throughput in function of the number of customers.

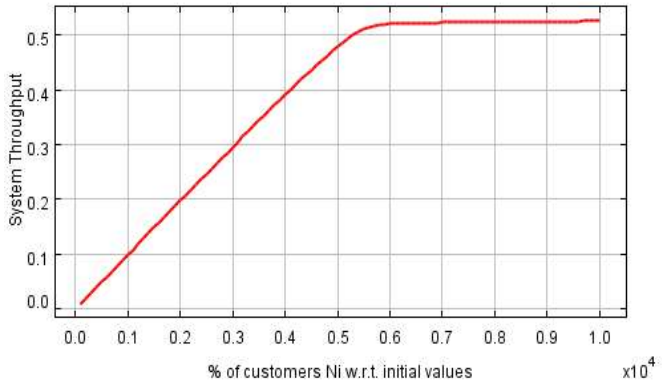

Fig. 9 System throughput

C. Use Machine Learning to Predict Server Utilization.

By using machine learning, we can accurately predict the server utilization on a given workload. Specifically, we can answer questions like: "***How much the server utilization will be if 500 users using the system concurrently?***"

To answer the previous question we used the technique of multi-linear regression. MLR is one of the machine learning prediction algorithm used to predict numerical values. Table 2 shows the sample dataset for database server utilization. It contains number of users, system response time, system throughput, web server utilization, application server utilization, and database server utilization.

TABLE 2
APPLICATION SERVER UTILIZATION FOR REGRISSION MODEL TRAINING DATASET SAMPLE

#users	SysRes	SystThro	webUtli	AppUtli	DbUtli
10	1008.25	0.01	0.01	0.02	0.02
69	1009.16	0.07	0.09	0.13	0.11
128	1010.23	0.13	0.17	0.24	0.21
200	1012.04	0.20	0.26	0.38	0.32
413	1025.03	0.40	0.53	0.77	0.66
489	1041.00	0.47	0.62	0.90	0.77
500	1045.12	0.48	0.63	0.92	0.78
572	1106.41	0.52	0.68	0.99	0.84
649	1243.86	0.52	0.68	1.00	0.86
899	1714.08	0.52	0.68	1.00	0.88
1340	2542.67	0.53	0.68	1.00	0.89
35	1008.54	0.03	0.05	0.07	0.06
2740	5169.85	0.53	0.69	?	0.91
433	1027.88	0.42	0.55	?	0.69
520	1055.05	0.49	0.65	?	0.80
600	1152.79	0.52	0.68	?	0.85
710	1358.83	0.52	0.68	?	0.86
1000	1904.06	0.53	0.68	?	0.88
5000	9408.67	0.53	0.69	?	0.92

D. Use the Machine Learning to Predict Risk Level

By using machine learning we can predict the level of risk whether it is low or medium or high risk, if we apply the software system on specific scenario. We can answer questions such as: "***What is the class/category of risk if 2000 users using the system concurrently?***"

Table 3 shows the training dataset for risk level prediction. It contains number of users, system response time, system throughput, web server utilization, application server utilization, database server utilization and class of the risk.

To answer the previous question we apply new instances as test set to predict the level of the risk. To enhance our work we apply and compare between three machine learning algorithms; Naïve Bays, K-Nearest Neighbor (KNN), and Support Vector Machines (SVMs).

TABLE 3
RISK LEVEL PREDICTION TRAINING DATASET

#users	SysRes	SystThro	webUtli	AppUtli	DbUtli	class
10	1008.25	0.01	0.01	0.02	0.02	LR
139	1010.50	0.14	0.18	0.26	0.22	LR
244	1013.47	0.24	0.32	0.46	0.39	LR
516	1052.77	0.49	0.64	0.94	0.80	MR
572	1106.41	0.52	0.68	0.99	0.84	MR
623	1195.13	0.52	0.68	1.00	0.85	HR
960	1828.90	0.52	0.68	1.00	0.88	HR
134	1010.37	0.13	0.17	0.25	0.22	LR
3700	6970.57	0.53	0.69	1.00	0.92	HR
4890	9408.67	0.53	0.69	1.00	0.92	HR
140	1010.51	0.14	0.18	0.27	0.23	LR
167	1011.14	0.17	0.22	0.32	0.27	LR
518	1053.88	0.49	0.64	0.94	0.80	?
5000	9408.67	0.53	0.69	1.00	0.92	?
1899	3591.77	0.53	0.69	1.00	0.91	?
710	1358.83	0.52	0.68	1.00	0.86	?
567	1099.52	0.52	0.68	0.99	0.84	?
33	1008.59	0.03	0.04	0.06	0.05	?
80	1009.33	0.08	0.10	0.15	0.13	?

III. RESULTS AND DISCUSION

We have presented the results of our experiments after using regression algorithm in prediction database server utilization. In addition, we used three classification algorithms in order to classify the level of risks whether it is high, medium, or low risk. At the end, we make a comparison between the algorithms we used.

A. Database server utilization prediction

We used WEKA as machine learning tool in order to predict database server utilization. Fig.10. represents the regression algorithm prediction with mean square error 0.01.

=== Evaluation on test split ===	
=== Summary ===	
Correlation coefficient	0.99
Mean absolute error	0.0067
Root mean squared error	0.0139
Total Number of Instances	194

Fig. 10 Prediction accuracy of database server utilization

B. Risk Level Classification

After prediction and using of server utilization database, next step is to classify the level of the risk. In addition, we compare between three machine learning algorithms Naïve Bayes, K-Nearest Neighbor (KNN), and Support Vector Machine (SVM).

1) Naïve Bayes: In Fig.11 we presented the accuracy of our approach by using Naïve Bayes algorithm. The figure stated that mean square error is 0.23.

=== Evaluation on test split ===		
=== Summary ===		
Correctly classified Instances	53	91.3 %
Incorrectly Classified Instances	5	8.6 %
Mean absolute error	0.06	
Root mean squared error	0.23	
Total Number of Instances	58	

Fig. 11 Naïve Bayes risk classification accuracy

Fig.12 stated the confusion matrix where 4 instances are classified as medium risk while they are high risk, and 1 instance is classified as medium risk while it is a high risk.

=== Confusion Matrix ===			
a	b	c	
22	4	0	a = LR
0	12	0	b = MR
0	1	19	c = HR

Fig. 12 Naïve Bayes risk classification confusion matrix

2) K-Nearest Neighbor (KNN): Fig.13 presented the accuracy percentage of KNN algorithm with mean square error 0.21.

=== Evaluation on test split ===		
=== Summary ===		
Correctly classified Instances	54	93.1 %
Incorrectly Classified Instances	4	6.8 %
Mean absolute error	0.05	
Root mean squared error	0.21	
Total Number of Instances	58	

Fig. 13 KNN risk classification accuracy

In Fig. 14 the confusion matrix of KNN stated that 3 instances are low risk while they classified as medium risk, and 1 instance is classified as medium risk while it is high risk.

=== Confusion Matrix ===			
a	b	c	
23	3	0	a = LR
0	12	0	b = MR
0	1	19	c = HR

Fig. 14 KNN risk classification confusion matrix

3) Support Vector Machine (SVM): In Fig.15 we presented the accuracy of SVM for prediction of risk classification. The mean square error is 0.4.

=== Evaluation on test split ===		
=== Summary ===		
Correctly classified Instances	40	68.9 %
Incorrectly Classified Instances	18	31.03 %
Mean absolute error	0.31	
Root mean squared error	0.40	
Total Number of Instances	58	

Fig. 15 SVM risk classification accuracy

In Fig.16 the confusion matrix of SVM showed that 2 instances are classified as medium risk while they are low risk, 4 instances are classified as high risk while they are low risk, and 12 instances are classified as high risk while they are medium risk

=== Confusion Matrix ===			
a	b	c	
20	2	4	a = LR
0	0	12	b = MR
0	0	20	c = HR

Fig. 16 SVM risk classification confusion matrix

The results of the experiments are compared in Table 4. The performances of the three models where evaluated based on three criteria, the prediction accuracy, learning time and error rate.

TABLE 4
COMPARISON BETWEEN CLASSIFIERS

Evaluation criteria	**Classifiers**		
	Naïve Bayes	**KNN**	**SVM**
Timing to build model (in sec)	0.01	0.01	0.28
Correctly classified instances	53	54	40
Incorrectly classified instances	5	4	18
Prediction accuracy	91%	93.1%	68.9%

The results indicate that the K-Nearest Neighbor classifier outperforms in prediction than Naïve Bayes and Support Vector Machine methods. Although, timing to build the model between Naïve Bayes and KNN is similar but, prediction accuracy differs significantly.

IV. Conclusion

In this paper, we have proposed model-based resource utilization and performance risk prediction using machine learning techniques approach. The performance risk can be categorized as time-related, or resource related performance risk. The approach starts by formulating UML diagrams, then mapping these diagrams into QNM model. We generate training dataset from QNM model by changing workloads. At the end we use this dataset to predict resource utilization and performance risk level.

The value of the work that it will provide an important feedback for software engineers before the software will be run in the working environment. Furthermore, it doesn't require deep knowledge from software engineer on QNM Model. Due to these reasons, our approach is suitable for analysis of performance risk at early phases of the software development life cycle.

In the future work the approach can be totally or partially automated to enable software engineer to take early decisions on building software easily starting from converting UML into QNM up to generating performance datasets.

ACKNOWLEDGMENT

This work was made possible by NPRP grant # [7-662-2-247] from Qatar Research Fund (a member of Qatar Foundation). The findings achieved herein are solely the responsibility of the authors.

REFERENCES

[1] V. Cortellessa, K. Goseva-popstojanova, S. S. Member, K. Appukkutty, A. R. Guedem, A. Hassan, S. S. Member, R. Elnaggar, W. Abdelmoez, S. S. Member, H. H. Ammar, and I. C. Society, "Model-Based Performance Risk Analysis," vol. 31, no. 1, pp. 3–20, 2005.

[2] A. Radhakrishnan and W. Virginia, "Tool Support for Software Performance Risk Assessment Tool Support for Software Performance Risk Assessment," 2007.

[3] H. A. Moniem, "A framework for Performance Prediction of Service-Oriented Architecture," vol. 4, no. 11, pp. 865–870, 2015.

[4] B. Islam, "Predict Software Reliability by Support Vector Machine," vol. 2, no. 4, pp. 46–52, 2013.

[5] B. Rabta, A. Alp, and G. Reiner, "Queueing Networks Modeling Software for Manufacturing."

[6] Z. Omary and F. Mtenzi, "Machine Learning Approach to Identifying the Dataset Threshold for the Performance Estimators in Supervised Learning," *Int. J.*, vol. 3, no. 3, pp. 314–325, 2010.

[7] Z. Omary and F. Mtenzi, "Dataset threshold for the performance estimators in supervised machine learning experiments," *2009 Int. Conf. Internet Technol. Secur. Trans.*, vol. 3, no. 3, pp. 314–325, 2009.

[8] Q. Zhang, L. Cherkasova, and E. Smirni, "A Regression-Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications," 2007.

[9] R. Mohanty, "Classification of Web Services Using Bayesian Network," *J. Softw. Eng. Appl.*, vol. 05, no. 04, pp. 291–296, 2012.

[10] N. Based, "Neighbor Based Algorithm for Multi-label Classification," pp. 718–721, 2004.

[11] S. Abe, "Fuzzy support vector machines for multilabel classification," *Pattern Recognit.*, vol. 48, no. 6, pp. 2110–2117, 2015.

[12] A. Ganapathi, "Predicting and optimizing system utilization and performance via statistical machine learning," *UC Berkeley*, no. UCB/EECS-2009–181, p. 97, 2009.

[13] K. Singh, E. \.Ipek, S. a McKee, B. R. de Supinski, M. Schulz, and R. Caruana, "Predicting parallel application performance via machine learning approaches:

Research Articles," *Concurr. Comput. Pr. Exper.*, vol. 19, no. 17, pp. 2219–2235, 2007.

[14] E. Ipek, B. R. De Supinski, M. Schulz, E. Ipek, B. R. de Supinski, B. R. de Supinski, M. Schulz, M. Schulz, S. A. McKee, and S. A. McKee, "An Approach to Performance Prediction for Parallel Applications," *Euro-Par 2005 Parallel Process.*, pp. 196 – 205, 2005.

[15] A. Brunnert and H. Krcmar, "Continuous performance evaluation and capacity planning using resource profiles for enterprise applications," *J. Syst. Softw.*, 2015.

[16] S. Balsamo, R. Mamprin, and M. Marzolla, "Performance evaluation of software architectures with queuing network models'," *Work. Softw. Perform.*, 1998.